# 19 sept.

\* Old Text

Nous avons remarqué que sur les 123 têtes transdisciplinaires, 86 % appartiennent au lexique transdisciplinaire des écrits scientifiques relevé par Tutin (2008).

de phénomènes de

ainsi que la présence la préposition

en élargissant à toutes les prépositions d’un côté mais en contraignant sur l’emplacement dans le titre. Dans le premier la tête transdisciplinaire est le premier nom du titre, dans le second elle est le premier nom après une marque de ponctuation, potentiellement une marque de segmentation, ce qui ferait de la tête transdisciplinaire le premier nom d’un second segment.

Les motifs fréquents dont sont issus ces schémas ont un faible le taux de croissance par rapport aux motifs des têtes non transdisciplinaires : ils ne sont pas spécifiques aux têtes transdisciplinaires. Ce fait s'explique car ce qui apparaît formellement comme une complémentation de nom est accessible à une grande quantité de noms, sinon tous, alors que

Projeter le premier schéma sur notre corpus retourne.

Nous construisons un lexique de ces noms en projetant nos deux schémas sur notre corpus de titre. Nous nous heurtons néanmoins à la difficulté que nous avons de trop nombreux résultats.

La recherche du schéma 1 sur notre corpus donne 39 729 couples (TRANS, NC) différents. Pour analyser s’il pourrait s’agir d’un emploi NSS nous décidons de nous concentrer sur une tête transdisciplinaire problème.

et plus particulièrement dans le schéma 1

Mais, en prévision d’un faible nombre de résultats, nous décidons de recherche notre schéma comme une séquence : on autorise une correspondance disjointe des énoncés avec le schéma, cela permet de maximiser la correspondance avec des variantes que l’on n’aurait pas prévues dans la définition initiale : par exemple, avoir un adjectif pour la tête transdisciplinaire, ce qui est une des séquences qui avait été trouvées lors de notre fouille de données séquentielles.

sur l’ensemble de notre corpus de travail. Sur les 94 têtes transdisciplinaires relevées

\* Old Code

reload(wb) ; cpt, heads = wb.recount\_transhead(titles)

for k, v in heads.items():

for t in trans:

if k == t.lemma:

if v != t.nb\_head['all']:

print(k, 'nb recount:', v, 'nb OneSegNoun:', t.nb\_head['all'])

break

OneSegNoun et mon recomptage tardif ne sont pas d’accord :

approche nb recount: 3414 nb OneSegNoun: 3422 +8

effet nb recount: 2093 nb OneSegNoun: 2094 +1

traitement nb recount: 449 nb OneSegNoun: 450 +1

WTF MAIS OUI ! OneSegNoun FAIT DE LA CORRECTION !!!

En reprenant les corrections (changement de lemme + de pos) j’arrive à :

**94739 TT par OneSegNoun**

Avec approche nb recount: 3443 nb OneSegNoun: 3422 -21

Si je ne corrige pas approche nb recount: 3414 nb OneSegNoun: 3422 : +8 🡺 **94 731**

Diff de 8 approches…

def count(d, k):

if k in d:

d[k] += 1

else:

d[k] = 1

# reload(wb) ; wb.xxtest2()

def xxtest2():

ponct = MockWord('PONCT', ':')

tt = MockWord('NC', 'problème')

p = MockWord('P', 'de')

nc2 = MockWord('NC', 'action')

t = MockTitle([ponct, tt, p, nc2])

r = is\_second\_ddaa(t)

print('Res:')

for e in r:

print(" " + str(e))

# reload(wb) ; wb.xxtest()

def xxtest():

nc1 = MockWord('NC', 'nc1')

ponct = MockWord('PONCT', ':')

tt = MockWord('NC', 'problème')

p = MockWord('P', 'de')

nc2 = MockWord('NC', 'nc2')

t = MockTitle([nc1, ponct, tt, p, nc2])

r = is\_second(t)

print('Res:')

for e in r:

print(" " + str(e))

def is\_second(t):

nc1 = None

ponct = None

tt = None

p = None

nc2 = None

for w in t.words:

#print(nc1, ponct, tt, p, nc2)

if ponct is None and w.pos == 'NC' and w.lemma not in TRANS:

nc1 = w

elif nc1 is not None and ponct is None and w.pos == 'PONCT':

ponct = w

elif ponct is not None and tt is None and w.pos == 'NC' and w.lemma in TRANS:

tt = w

elif tt is not None and w.pos in ['P', 'P+D']:

p = w

elif p is not None and w.pos == 'NC' and w.lemma not in TRANS:

nc2 = w

break

if nc1 is not None and ponct is not None and tt is not None and p is not None and nc2 is not None:

break

return(nc1, ponct, tt, p, nc2)

def is\_first(t):

x = []

if i not in [0, 1] or t.words[i].lemma not in TRANS: return None, None, None

for j in range(i, len(t.words)):

if j == i:

x.append(t.words[j]

elif j == i + 1 and 1

if i in [0, 1]:

if root1.lemma in TRANS:

if i + 1 < len(t.words):

if t.words[i + 1].pos == 'ADJ':

if i + 2 < len(t.words):

if t.words[i + 2].pos == 'P' and t.words[i + 2].lemma in ['à', 'sur', 'de']:

if i + 3 < len(t.words):

if t.words[i + 3].pos == 'DET':

if i + 4 < len(t.words):

t.words[i + 4].pos == 'NC':

elif t.words[i + 1].pos == 'P

def is\_first(t):

tt = None

p = None

nc = None

# first pattern

for i, w in enumerate(t.words):

if i == 0:

if w.pos == 'DET' and i + 1 < len(t.words):

if t.words[i + 1].pos == 'NC' and t.words[i + 1].lemma in TRANS:

tt = t.words[i + 1]

elif w.pos == 'NC' and w.lemma in TRANS:

tt = w

elif tt is not None and w.pos in ['P', 'P+D']:

p = w

elif p is not None and w.pos == 'NC': # and w.lemma not in TRANS:

nc = w

if tt is not None and p is not None and nc is not None:

break

return (tt, p, nc)

18 sept.

\* Old Text

Elles couvrent 93 457 titres, soit 37 % des titres de notre corpus.

Autre information que nous obtenons en faisant un comptage sur notre corpus

, comme elle est en tête de premier segment avec le motif 4, INIT TransHead

Les têtes transdisciplinaires sont la tête, 9 595 premiers segments de titres bisegmentaux,

 13

mais cette problématique vaste s’éloigne trop de notre sujet.

imposerait un sujet qui, s’il ne peut pas déduit du contexte, devrait être le plus neutre possible.

Dans notre exemple, la paraphrase en infinitive perd le sujet de sélectionne dans la conjonctive, le jury.

et nombreuses

la plus restreinte possible

se pencher sur les conséquences

définitions de

des CS décrites, dites classiques désormais.

est inclus dedans et

que l’on peut voir comme

et passer de l’une à l’autre en paraphrasant

signale toutefois la différence majeure

La définition de la proposition On peut donc se demander dans

Cette obligation d’avoir un sujet propre, différent de la principale, potentiellement implicite et impersonnel comme dans l’exemple de CS-II, n’est pas forcément toujours respectée. D’où la seconde hypothèse de considérer le contenu spécifiant comme un syntagme prépositionnel introduit par de incluant un syntagme verbal dont le verbe est à l’infinitif.

, considérant ici le de comme un subordonnant équivalent à que, rejoint en cela par La paraphrase de l’infinitive vers la conjonctive demanderait de réintroduire un sujet pour que la subordonnée produite soit grammaticale et acceptable.

Nous évitons le terme de complétive car, s’il signifie que la proposition peut occuper les fonctions d’un nom, il se rapporte directement au nom d’une fonction, celle de complément, alors qu’un nom, et d’autant plus les constructions spécificationnelles avec le verbe être, rapproche le contenu spécifiant de la fonction d’attribut ou complément attribut selon la terminologie de Delhay (2014).

qui sont les plus grands conteneurs de séquences

Nous faisons cela pour toutes les têtes nominales, transdisciplinaires ou non transdisciplinaires, de nos titres.

dont les 7 443 séquences ont pour pivot une tête transdisciplinaire et, de l’autre, la base commune dont les 23 970 séquences ont pour pivot une tête non transdisciplinaire.

Nous nous intéressons à l’environnement immédiat des têtes  et

et d’autres pareillement problématique.

, soit 74, ce qui sélectionne 757 motifs

nous donne plus d’information sur ce qui suit les séquences où les têtes transdisciplinaires sont en tête de segment : un déterminant qui, selon la syntaxe, doit être suivi par un nom (motifs 1, 2, 3). On le voit déjà apparaître lorsque le déterminant est fusionné avec la préposition (motif 4).

Figure 1 :  Treillis des séquences les plus fréquentes des têtes transdisciplinaires

Le treillis représenté dans la figure (1) permet d'avoir un aperçu de ces séquences, une flèche indiquant la relation "est contenu dans" qui est transitive (pour trois séquences A, B et C, si A -> B -> C alors A -> C). Par souci de lisibilité, nous n'avons pas fait figurer les relations déductibles par transitivité. Nous avons étagé le diagramme selon le nombre d'éléments par séquence et nous avons filtré pour ne garder que les séquences les plus fréquentes à chaque niveau.

Normalement, il n'y aucune relation sautant un niveau, néanmoins avec les filtres appliqués sur les fréquences, certaines séquences existantes mais pas assez fréquentes ne sont pas affichées. Nous avons donc fait figurer en rouge les relations qui sautent des niveaux de l'arbre.

La fouille de données séquentielles est une méthode qui peut être appliquée de façon neutre à n'importe quel type de données. Nos connaissances linguistiques nous permettent d'interpréter ce diagramme, notamment l'optionnalité de certains items,

B) Deux schémas fréquents proches de la CS-VII

Si le taux de croissance ne donne rien de probant, cela signifie que les têtes transdisciplinaires ne se distinguent pas syntaxiquement des têtes non transdisciplinaires. Néanmoins, nous pouvons utiliser le comptage des séquences pour faire émerger les motifs les plus fréquents des têtes transdisciplinaires sur les 1 604 847 séquences recensées.

Ils apportent comme informations supplémentaires que la séquence PONCT TransHead à

Nous calculons ensuite le taux de croissance de chaque motif transdisciplinaire par rapport au motif correspondant dans l’autre base : nous calculons ainsi le taux de croissance du motif transdisciplinaire PONCT **TT** de par rapport au motif commun PONCT **NC** de. S’il n’y a pas de motif commun correspondant, le taux de croissance est infini. Sinon il est égal au support de la séquence transdisciplinaire divisé par le support de la séquence non transdisciplinaire.

Les taux de croissance des 757 motifs se divisent en deux catégories : 33 motifs ont un taux de croissance égale à l’infini et les 724 autres ont  un taux de croissance inférieur à 1, ils ne sont donc pas émergents. On remarque que le motif avec le plus haut taux de croissance, 0,78 ,

Néanmoins, nos résultats ne permettent pas de distinguer des motifs émergents propres aux têtes transdisciplinaires par rapport aux motifs des têtes non transdisciplinaires.

FAUX

Pour préparer ce retour vers des schémas linguistiques, nous comptes également les occurrences des séquences dans les titres : ainsi, la séquence **DET** **TT** aura ainsi une occurrence dans **DET** **TT** de mais pas dans **DET** NC PONCT **TT** ni **DET** ADJ **TT**. C’est une manière de calculer la fréquence des séquences dans les titres en activant la contrainte de l’ordre des items mais aussi la contrainte la contiguïté des items. Ce calcul ne se fait pas non plus en prenant en compte des informations linguistiques. On pourrait en effet rapprochait les séquences **DET** **TT** et  **DET** ADJ **TT** en considérant que l’adjectif est un mot optionnel, comme nous le ferons par la suite.

\* Old code

Pour le calcul des motifs, une partie a été intégré à datamodel, l’autre est là :

# Update the 17/09

# motifs(titles, -1, +1) : on va chercher les motifs A head C

# on ne garde les lemmes que pour les classes fermées DET P P+D CS CC PROREL et être et avoir sinon POS

# reload(wb) ; test = { '62226' : titles['62226'] } ; ngss, nc = wb.motifs(test, 2, 2)

# lemma::pos

# ou pos (y compris INIT et END)

# ('INIT', 'NGSS') => ('INIT', 'NC')

def fouille\_change(key):

    neo = []

    for item in key:

        if item == 'NGSS':

            neo.append('NC')

        else:

            neo.append(item)

    return tuple(neo)

# reload(wb) ; test = { '62226' : titles['62226'] } ; ngss, nc = wb.motifs(test, 2, 2) ; ngss[('DET', 'NGSS', 'V')] = 1 ; wb.support(ngss)

# L'égalité peut avoir des trous : (A, C) est contenue dans (A, B, C) <=> (A, B, C) est une sous-séquence de (A, B)

# Donc on n'utilise plus ça qui était une égalité stricte

#def cmp\_seq(seq1, seq2):

#    maxx = min(len(seq1), len(seq2))

#    for i in range(0, maxx):

#        if not cmp\_item(seq1[i], seq2[i]):

#            return False

#    return True

class Seq:

    def \_\_init\_\_(self, tab):

        self.length = tab[0]

        self.items = tab[1:6]

        self.count = tab[6]

    def \_\_str\_\_(self):

        s = '(' + str(self.length) + ') '

        for i in range(0, self.length):

            s += self.items[i] + ' '

        return s

    def \_\_repr\_\_(self):

        return str(self)

    def contains(self, s):

        return is\_contained(self, s)

# reload(wb) ; wb.fouille\_test()

# Check support

# reload(wb) ; wb.count\_one('PONCT')

def count\_one(val):

    wb = load\_workbook(r".\output\motifs\_supports-2-7.xlsx", read\_only=True)

    pct = 0

    for i, sn in enumerate(wb.sheetnames):

        ws = wb[sn]

        if i == 0:

            for row in ws.rows:

                for j in range(1, 6):

                    lem = row[j].value

                    if lem == val:

                        pct += 1

    return pct

def read(filename, sheet):

    wbin = load\_workbook(filename, read\_only=True)

    for i, sn in enumerate(wbin.sheetnames):

        ws = wbin[sn]

        if i == sheet:

            lines = []

            for row in ws.rows:

                line = []

                for cell in row:

                    line.append(cell.value)

                lines.append(line)

    return lines

# [ '', 'A', 'B', '\_', '\_', '\_'] => ['A', 'B']

def grep\_motif(line):

    motif = []

    for k in line[1:6]:

        if k != '\_':

            motif.append(k)

        else:

            return motif

    return motif

# reload(wb) ; wb.test\_count\_suite()

def count\_suite(motif, data, exclude\_first=True):

    pct = 0

    for row in data:

        if exclude\_first:

            exclude\_first = False

            continue

        #print('INFO', len(row), row)

        good = 0

        for j in range(1, 6):

            #print(motif[0], j, row[j])

            if exq(motif[0], row[j]):

                good = 0

                for k in range(j, j + len(motif) + 2):

                    if k < len(row) and k < 6 and exq(motif[good], row[k]):

                        #print('k', k, 'D', row[k], 'M', motif[good], '+1')

                        good += 1

                        if good == len(motif):

                            break

                    #else:

                    #    print('k', k, 'D', row[k], 'M', motif[good])

                if good == len(motif):

                    break

        #print(row)

        if good == len(motif) and not strict\_seq\_eq(motif, row[1:6]): #(because of identity)

            pct += 1

            #print('good')

        #elif strict\_seq\_eq(motif, row):

        #    print('equal')

        #print()

    return pct

# r".\output\motifs\_supports-2-7.xlsx"

# reload(wb) ; r = wb.recalc\_support(r".\output\motifs\_supports-2-5.xlsx")

def recalc\_support(filename):

    # read

    print('READING')

    ngss = read(filename, 0)

    nc = read(filename, 1)

    # recalc support nc

    correponding = {}

    print('RECALC NC')

    first = True

    # count

    cpt = 0

    seuil = 1000

    total = 0

    for line in nc:

        if first:

            first = False

        else:

            #print(line, grep\_motif(line))

            try:

                nc[7] = count\_suite(grep\_motif(line), nc) / (len(nc) - 1)

                # NONONONONO

                key = line[1].replace('NC', 'NGSS') + line[2].replace('NC', 'NGSS') + line[3].replace('NC', 'NGSS') + line[4].replace('NC', 'NGSS') + line[5].replace('NC', 'NGSS')

                corresponding[key] = cpt

            except AttributeError:

                pass

        cpt += 1

        total += 1

        if cpt == seuil:

            print(f"{total:10d} / {len(nc)-1:10d}")

            cpt = 0

    # recalc support ngss

    print('RECALC NGSS')

    first = True

    # count

    cpt = 0

    seuil = 1000

    total = 0

    for line in ngss:

        if first:

            first = False

        else:

            #print(line, grep\_motif(line))

            try:

                ngss[7] = count\_suite(grep\_motif(line), ngss) / (len(ngss) - 1)

                key = line[1] + line[2]+ line[3] + line[4] + line[5]

                if key in corresponding:

                    ngss[8] = ngss[7] / nc[corresponding[key]][7]

                else:

                    ngss[8] = '∞'

            except AttributeError:

                pass

        cpt += 1

        total += 1

        if cpt == seuil:

            print(f"{total:10d} / {len(ngss)-1:10d}")

            cpt = 0

    # output

    wbout = openpyxl.Workbook(write\_only=True)

    ws = wbout.create\_sheet('Motifs Supports NGSS')

    #ws.append(['Len', '1', '2', '3', '4', '5', 'Count', 'Support', 'Croissance'])

    for line in ngss:

        ws.append(line)

    ws = wbout.create\_sheet('Motifs Supports NC')

    ws.append(['Len', '1', '2', '3', '4', '5', 'Count', 'Support', 'Croissance'])

    for line in nc:

        ws.append(line)

    wbout.save('motifs\_supports\_c.xlsx')

    return ngss, nc

# reload(wb) ; wb.f\_motifs\_supports(ngss, nc, supports\_ngss, supports\_nc)

Pour le recalcul des motifs, le code qui restait (non repris) :

#-----------------------------------------------------------

# MOTIFS

#-----------------------------------------------------------

def item2lem\_pos(item):

    if len(item.split('::')) == 1: # pos

        lem = 'SPECIAL\_ANY'

        pos = item

    else: # lemma::pos

        lem, pos = item.split('::')

    return lem, pos

# pos is stronger for PRIME ONLY! (if no lemma is defined for only one item1, no comparison is made on it)

def cmp\_item(item1, item2):

    lem1, pos1 = item2lem\_pos(item1)

    lem2, pos2 = item2lem\_pos(item2)                      # séquence vs sous-séquence

    if lem1 == 'SPECIAL\_ANY' and lem2 == 'SPECIAL\_ANY':   #     (V) vs (V)

        return pos1 == pos2

    elif lem1 == 'SPECIAL\_ANY' and lem2 != 'SPECIAL\_ANY': #      (V) vs (être V)

        return True

    elif lem1 != 'SPECIAL\_ANY' and lem2 == 'SPECIAL\_ANY': # (être V) vs (V)

        return False

    else:                                                 # (être V) vs (être V)

        return lem1 == lem2 and pos1 == pos2

# S'est contenu dans S ? <=> S' est une FORME PLUS GENERIQUE de S

# S' = <(DET) (NC) ,(être V)>

# S = <(le DET) (solution NC) (être V) (de P)>

# S' est contenue dans S

# S est une sous-séquence de S'

# S <\_ S'

def is\_contained(prime, s):

    #print(prime, 'vs', s)

    if len(prime) > len(s): return False # S' doit avoir une longueur plus petite ou égale à S

    nb = 0

    for item in s:

        #print('   ', nb, prime[nb], 'vs', item, cmp\_item(prime[nb], item))

        if cmp\_item(prime[nb], item):

            nb += 1

            if nb == len(prime): # tu restarts pas à 0 !!! c mauvais... voir def count\_suite

                return True

    return False

import sys

def ptest(expr, val):

    if expr != val:

        sys.stderr.write(str(expr) + ' vs expected ' + str(val) + '\n')

    else:

        print(expr, ' (expected ', val, ')', sep='')

# reload(wb) ; wb.test\_motifs()

def test\_motifs():

    s1 = ('DET', 'ADJ', 'NGSS')

    prime1 = ('DET', 'NGSS')

    ptest(is\_contained(prime1, s1), True)

    s2 = ('de::DET', 'ADJ', 'évolution::NGSS')

    prime2 = ('DET', 'NGSS')

    ptest(is\_contained(prime2, s2), True)

    s3 = ('DET', 'NC', 'être::V')

    prime3 = ('DET', 'NC', 'V')

    ptest(is\_contained(prime3, s3), True)

    s4 = ('DET', 'NC', 'V')

    prime4 = ('DET', 'NC', 'être::V') # False : prime4 is more precise than s4!

    ptest(is\_contained(prime4, s4), False)

#---------------------------------------------------

# r".\output\motifs\_supports-2-7.xlsx"

# reload(wb) ; r = wb.recalc\_support(r".\output\motifs\_supports-2-5.xlsx")

def recalc\_support(filename):

    # read

    print('READING')

    ngss = read(filename, 0)

    nc = read(filename, 1)

    # recalc support nc

    correponding = {}

    print('RECALC NC')

    first = True

    # count

    cpt = 0

    seuil = 1000

    total = 0

    for line in nc:

        if first:

            first = False

        else:

            #print(line, grep\_motif(line))

            try:

                nc[7] = count\_suite(grep\_motif(line), nc) / (len(nc) - 1)

                # NONONONONO

                key = line[1].replace('NC', 'NGSS') + line[2].replace('NC', 'NGSS') + line[3].replace('NC', 'NGSS') + line[4].replace('NC', 'NGSS') + line[5].replace('NC', 'NGSS')

                corresponding[key] = cpt

            except AttributeError:

                pass

        cpt += 1

        total += 1

        if cpt == seuil:

            print(f"{total:10d} / {len(nc)-1:10d}")

            cpt = 0

    # recalc support ngss

    print('RECALC NGSS')

    first = True

    # count

    cpt = 0

    seuil = 1000

    total = 0

    for line in ngss:

        if first:

            first = False

        else:

            #print(line, grep\_motif(line))

            try:

                ngss[7] = count\_suite(grep\_motif(line), ngss) / (len(ngss) - 1)

                key = line[1] + line[2]+ line[3] + line[4] + line[5]

                if key in corresponding:

                    ngss[8] = ngss[7] / nc[corresponding[key]][7]

                else:

                    ngss[8] = '∞'

            except AttributeError:

                pass

        cpt += 1

        total += 1

        if cpt == seuil:

            print(f"{total:10d} / {len(ngss)-1:10d}")

            cpt = 0

    # output

    wbout = openpyxl.Workbook(write\_only=True)

    ws = wbout.create\_sheet('Motifs Supports NGSS')

    #ws.append(['Len', '1', '2', '3', '4', '5', 'Count', 'Support', 'Croissance'])

    for line in ngss:

        ws.append(line)

    ws = wbout.create\_sheet('Motifs Supports NC')

    ws.append(['Len', '1', '2', '3', '4', '5', 'Count', 'Support', 'Croissance'])

    for line in nc:

        ws.append(line)

    wbout.save('motifs\_supports\_c.xlsx')

    return ngss, nc

# reload(wb) ; wb.count\_suite(['PONCT', 'NGSS', 'VINF'])

# reload(wb) ; wb.count\_suite(['PONCT', 'VINF'])

# 3  PONCT  NGSS VINF  \_ \_ 1 8610  ∞

# 4  VINF PONCT  NGSS VINF \_  1 774 ∞

# 5  PONCT  VINF PONCT  NGSS VINF 1  42 ∞

# 4  PONCT  NGSS VINF  en::P \_ 1 30  ∞

# 5  VINF PONCT  NGSS VINF en::P  1 0 ∞

# 5  PONCT  NGSS VINF  en::P NPP 1  0 ∞

# 6

# reload(wb) ; wb.test\_count\_suite()

def test\_count\_suite():

    data = [

            ['', 'PONCT', 'PIPO', 'VINF', '', ''], # PYTHON CONCAT LES STRINGS SANS RIEN !

            ['', 'PONCT', 'VINF', '', '', ''],

            ['', 'PONCT', 'ZORBA', 'DRACULA', 'VINF', ''],

            ['', 'VINF', 'PONCT', '', '', ''],

            ['', 'PONCT', 'PONCT', 'VINF', '', ''],

            ['', 'de::DET', 'NC', '', '', ''],

            ['', 'NC', 'V', '', '', '']

        ]

    motif = ['PONCT', 'VINF']

    print('Should be True :', strict\_seq\_eq(motif, motif))

    print('Motif:', motif)

    print('Should be 3 :', count\_suite(motif, data))

    motif = ['DET', 'NC']

    print('Motif:', motif)

    print('Should be 1 :', count\_suite(motif, data))

    motif = ['pipo::NC', 'V']

    print('Motif:', motif)

    print('Should be 0 :', count\_suite(motif, data))

def strict\_item\_eq(a, b):

    #print('==', a, b)

    return a == b

def strict\_seq\_eq(motif, sb):

    ok = True

    #print(len(motif), len(sb))

    if len(motif) > len(sb): return False

    for i in range(0, len(motif)):

        #print('=', 'i', i, 'mot', motif[i], 'row', sb[i], '.')

        if not strict\_item\_eq(motif[i], sb[i]):

            return False

    return ok

def count\_suite\_direct(vals):

    wb = load\_workbook(r".\output\motifs\_supports-2-7.xlsx", read\_only=True)

    pct = 0

    for i, sn in enumerate(wb.sheetnames):

        ws = wb[sn]

        if i == 0:

            for row in ws.rows:

                for j in range(1, 6):

                    lem = row[j].value

                    if lem == vals[0]:

                        #good = True

                        good = 0

                        for k in range(j, j + len(vals) + 1):

                            # non disjoint

                            #if k >= len(row) or row[k].value != vals[k - j]:

                            #    good = False

                            #    break

                            # disjoint

                            #print(i, k, len(row), row[k].value, good, vals[good])

                            if k < len(row) and row[k].value == vals[good]:

                                good += 1

                                if good == len(vals):

                                    break

                        #if good:

                        if good == len(vals):

                            pct += 1

                            #for i in range(0, len(row)):

                            #    print(row[i].value, ' ', end='')

                            #print('good')

    return pct

# 17 sept.

\* Old Code

# non used

def fouille\_filter(mtfs, value):

    res = {}

    for key in mtfs:

        if mtfs[key] >= value:

            res[key] = value

    return res

# tt, nc = last\_recalc("output\motifs\_supports-2-7.xlsx")

def last\_recalc(filename):

    wbin = load\_workbook(filename, read\_only=True)

    tt = []

    nc = []

    ws = wbin['Motifs Supports NGSS']

    first = True

    for row in ws.rows:

        if first:

            first = False

            continue

        data = []

        for cell in row:

            data.append(cell.value)

        if len(data) > 7:

            tt.append(Seq(data[:7]))

    ws = wbin['Motifs Supports NC']

    first = True

    for row in ws.rows:

        if first:

            first = False

            continue

        data = []

        for cell in row:

            data.append(cell.value)

        if len(data) > 7:

            nc.append(Seq(data[:7]))

    del wbin

    return tt, nc

    #first\_prime = prime[0]

    #for i, item in enumerate(s):

    #    if cmp\_item(first\_prime, item):

    #        if cmp\_seq(prime, s[i:]):

    #            return True

    #return False

#

# La classe Seq et last\_recalc sont les deux derniers éléments

# Ils utlisent is\_contained(prime, s) et cmp\_item(item1, item2)

#

\* Old Text

Chaque séquence est composée d’items qui sont, pour les classes fermées, le lemme du mot, et, pour les classes ouvertes, son étiquette morphosyntaxique, sauf pour les verbes être et avoir où nous gardons également le lemme.

à deux exceptions près. La première exception est être et avoir pour lesquels on prend le lemme et non l’étiquette, les distinguant ainsi des autres verbes.

La seconde exception est que l’on met l’étiquette NSS pour les têtes transdisciplinaires au lieu de NC pour nom commun.

CS pour conjonction de subordination,

, CC pour conjonction de coordination

: la fouille de données séquentielles

avec cette dernière séquence que la fouille de données séquentielles, applicable dans d’autres disciplines, ignorent les règles propres de la linguistique.

Le support d’une séquence S dans une base donnée est le nombre de séquences contenant S, c’est-à-dire qu’elles contiennent tous les items de S dans le même ordre, y compris de façon disjointe.

# 16 sept.

\* Old Text

La « titular colonicity » a été beaucoup étudié dans la littérature.

Il nous semble possible de rapprocher (1) et (3) et (2) et (4).

Comme nous l’expliquions dans notre travail de première année, c’est

fréquent pour les titres

Cela le schéma sémantique de présenter le concept général avant le cas ou l’exemple spécifique (, opérant un

On peut également étudier les titres bisegmentaux en prenant les deux têtes ensembles,

s’explique car nous exigeons qu’une des têtes soient un nom, l’autre peut être un verbe ou une préposition.

Cette préposition est utilisée dans des structures de la forme de … vers …, de … à … étudiées par Tanguy et Rebeyrolle (à paraître), mais comme il ne s’agit pas d’un nom nous l’écartons ici.

est immédiate pour (35)

# 15 sept.

\* Old Text

le cas nom-description

Tutin (2008) fixe la présence d’au moins 15 occurrences d’un lemme dans les trois domaines qu’elle étudie pour la qualifier de transdisciplinaire.

’une forme dans 15 domaines comme marque de sa transdisciplinarité.

15 domaines représentent 60 % des 25 domaines retenus pour nos calculs sur les 27 de notre corpus.

ce qui peut expliquer l’absence du lemme modélisation qui s’apqui dénote les sciences exactes

Pour les autres lemmes,

sommes partis quant à nous de l’hypothèse que

distinct du vocabulaire général.

, nous identifions En plus de la liste des 94 têtes

Sur les 123 têtes transdisciplinaires relevées, 103 sont également des têtes spécifiques, soit 84 %.

Néanmoins, l’ampleur du travail rend peu probable qu’il fasse l’objet d’une publication aussi courte qu’un article, un chapitre d’ouvrage, une communication ou un poster.

que l’on peut rapprocher des n. Certains peuvent néanmoins avoir deux emplois : outil par

que l’on peut rapprocher XXX

) reprend la distinction en trois de ordres de Lyons (1977) des entités dénotées par les noms. Nous traduisons ci-dessous la définition qu’en fait:

et avoir accepter les aménagements nécessaires à sa poursuite

qu’a duré ce projet.

ne lira ni le résumé ni l’article après avoir lu

Soit car le lecteur détermine que l’article ne l’intéresse pas, soit car il trouve ce qu’il cherche.

Ce dernier article indique même que certains médecins vont jusqu’à prendre des décisions cliniques fondées uniquement sur les titres.

en fonction de mesures

, par exemple,

Le premier schéma couvrait 50 % des titres, le deuxième 5 % et le dernier 10 %, soit une couverture totale de 65 % de notre corpus. Nous avions ensuite étudié les noms et les couples de noms les plus fréquents qui peuplaient ces schémas. Nous avions constaté

Une hypothèse est plus Utiliser un plus grand corpus permet une pluPlus il est large, plus il donne une fondation solide pour la confirmation ou l’infirmation d’hypothèses dessus.

cas les plus fréquents, les

Plus le nombre de titres étudiés est faible, plus les résultats sont très sensibles. Ils dépendent d’un faible nombre d’occurrences, ce qui ne les rend pas fiables car la prise en compte d’occurrences supplémentaires amènerait .

Cette très forte présence ne se rencontre pas dans l’ensemble du corpus et le corpus des premiers segments des titres bisegmentaux.

de travail

surreprésentées dans le sous-corpus des seconds segments de titres bisegmentaux

Les occurrences des têtes cas et exemple représentent 4 % pour cas et 3

Un fait remarquable du sous-corpus des seconds segments de titres bisegmentaux, c’est que certaines têtes transdisciplinaires sont surreprésentées par rapport aux autres sous-corpus.

Dans l’ensemble du corpus, cas ne représentent que 1,3 % des têtes,

En tout, les

De plus les têtes cas et exemple sont spécifiques au sous-corpus des seconds segments. Les occurrences de la tête étude du corpus de travail ne représente que 2 % du total des occurrences de têtes, celles des têtes analyse et étude près de 1 % du corpus des premiers segments des titres bisegmentaux. Dans le sous-corpus des titres monosegmentaux, la tête étude est à 3 %.

# 14 sept.

\* Vieux texte

de la publication titrée

de noms abstraits

Nous souhaitons tout d’abord étendre le périmètre de notre étude à l’ensemble des titres : ceux

Nous faisons l’hypothèse, soutenue par notre intuition et notre connaissance du précédent corpus, que le premier nom immédiatement après le double point que nous étudions est la tête, e, du syntagme de premier niveau du segment après le double point et donc la tête du segment. Nous redéfinissons notre cible d’étude comme les têtes de segments et nous élargissons cette étude, en ne regardant plus seulement le segment après le double point, mais aussi le segment avant.

Tout d’abord, Nous pensons que la classe de nom ayant émergé dans notre premier travail peut gagner à être redéfinie par une autre approche, indépendante de la position des noms immédiatement après le double point, à la fois plus stricte et d’une couverture plus large. Nous avions en effet écarté dans notre précédente étude toute la partie avant le double point et les phénomènes récurrents pouvant y survenir, perdant ainsi des découvertes potentielles et n’utilisant pas une large partie de notre corpus.

Nous élargissons également notre étude aux titres à un seul segment et aux titres à deux segments séparés par un autre signe de ponctuation que le double point. Notre étude portera donc sur toutes les têtes nominales des segments des titres à un ou deux segments.

Nous parlerons pour désigner cet emploi, en reprenant la terminologie de Legallois (2008), de noms sous-spécifiés (NSS).

La similitude lexicale

La question initiale est donc de savoir si nos noms seraient des NSS.

rapprochement possible avec les noms sous-spécifiés (NSS), reprenant la terminologie de Legallois (2008).

Pour cela, nous voulons caractériser ces têtes et les schémas récurrents dans lesquels elles s’insèrent, dans un corpus de titres de publications scientifiques. Nous voulons ensuite rapprocher les têtes des noms sous-spécifiés et les schémas des constructions spécificationnelles.

Ce sont ces têtes de segments dont nous voulons étudier l’éventuelle sous-spécification.